File System

File System

Milestone One

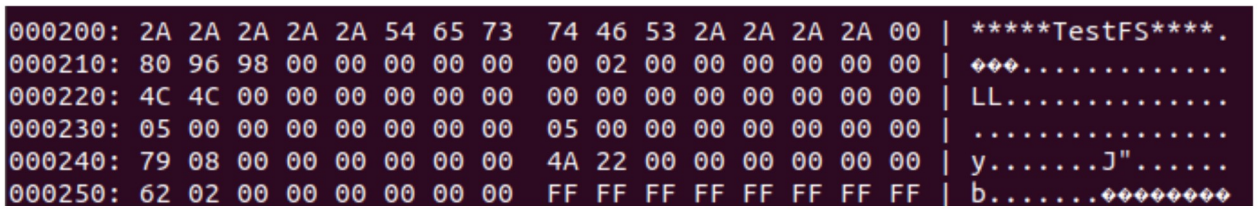
Team name: Team 05

Team member	Student ID	GitHub name
Bisum Tiwana	**920388011**	**SpindlyGold019**
Babak Milani	**920122577**	**babakmilani**
Gurpreet Natt	**922883894**	**gpreet2**
Mozhgan Ahsant	**921771510**	**AhsantMozhgan**

1. A dump (use the provided HexDump utility) of the volume file that shows the VCB, FreeSpace, and complete root directory.

a. Magic Number : 0x5A3C7F2A5A3C7F2A

In little-endian systems our magic number appears at 2A7F3C5A2A7F3C5A



b. Free Space Management:

File System

```
01010011 00101010 00101010 00101010 00101010
00000000 10000000 10010110 10011000 00000000
```

c. Root Directory

d. Generating a PDF Dump

To generate a PDF dump of the volume that includes the VCB, FreeSpace, and root directory, we can use the provided “hexdump” utility. We need to run this utility on our volume file and capture its output in a PDF file. Here's how we can do it:

- Build our project with the hexdump utility.

- Run the hexdump utility on our volume file: ./hexdump -c 0 -s 1 -f volume.dat > dump.txt

- Convert the dump.txt to a PDF file. We can use various tools or commands to convert text to PDF, such as a2ps and ps2pdf.

The resulting PDF should include the hexadecimal and ASCII representation of the VCB, FreeSpace, and root directory.

2. A description of the VCB structure?
The VCB is typically stored at the beginning of the volume (block 0). It contains essential information for managing the file system, such as the volume size, block size, free space status, and the location of the root directory. The Volume Control Block (VCB) is a crucial data structure in a file system that holds several variables.

a. Number of Blocks: It represents the total number of blocks in the volume, and provides an overview of the size of the storage available in your file system.
b. Block Size: It specifies the size of each block in bytes. Block size determines how much data can be stored in a single block.
c. Free Space:It stores information about the available free space in the volume. It indicates the number of free blocks that can be allocated for file storage.
d. Root Directory: It points to the block number where the root directory of the file system is located. The root directory is the top-level directory in your file system and serves as the starting point for all directory structures.
e. Free Space BitMap: It is a data structure that keeps track of the allocation status of each block in the volume. It is often implemented as a bitmap where each bit corresponds to a block. A value of 1 indicates an allocated block, while 0 represents a free block.
f. BitMap Byte Size: It specifies the size of the free space bitmap in bytes. It is necessary to manage the allocation of blocks efficiently.

File System

3. A description of the Free Space structure?
The Free Space structure is responsible for managing and tracking the allocation and deallocation of storage blocks within the file system. It plays a crucial role in ensuring that the file system efficiently uses available storage space. The Free Space structure ensures efficient utilization of storage space within the file system by keeping track of which blocks are available for allocation and which are already in use. It is a critical component for maintaining the file system's integrity and preventing data overwrites or corruption.

a. Allocation Procedure (allocate):The Free Space structure includes a procedure for allocating free blocks to store data. When a file is created or data is written to the file system, the allocate procedure is called to reserve free storage blocks for the file. This procedure updates the Free Space bitmap to mark the allocated blocks as in use.
b. Deallocation Procedure (release): a complete file system should also include a deallocation procedure. This procedure releases allocated blocks back into the pool of free space when files are deleted or storage space is no longer needed.
c. Free Space BitMap: The Free Space structure often includes a bitmap, known as the freeSpaceBitMap, which serves as a map of the storage blocks within the file system. Each bit in the bitmap corresponds to a storage block. A value of 1 typically indicates that the corresponding block is allocated (in use), while a value of 0 signifies that the block is free and available for allocation.
d. BitMap Byte Size: It specifies the size of the Free Space bitmap in bytes. It is essential for managing and manipulating the bitmap efficiently. The bitmap size is often calculated based on the total number of storage blocks in the file system.

4. A description of the Directory system?
The Directory system is responsible for organizing and managing files and directories within the volume. It provides a structured way to access, locate, and navigate through the files and directories stored on the volume. Below are the key components and concepts related to the Directory system:

a. Directory Entry: A directory entry represents an individual file or subdirectory stored within the parent directory. Each directory entry typically includes the following information:
- Filename: The name of the file or subdirectory. Filenames are usually limited to a certain length and may follow specific naming conventions.
- File Type: Indicates the type of the entry, such as regular file, directory, or symbolic link.
- Location: Stores information about where the file's data blocks are located on the storage volume.
- Size: Indicates the size of the file or subdirectory in bytes.
- Timestamps: Record important timestamps, including the last access time, modification time, and creation time.

b. Root Directory: The root directory is the top-level directory in the file system. It serves as the starting point for the entire directory hierarchy. In Unix-like file systems, it is represented as a forward slash ("/"). The root directory must be created and initialized during the formatting of the volume and typically contains at least two special entries:

- ".": This entry represents the directory itself, allowing navigation within the directory hierarchy.
- "..": This entry represents the parent directory, enabling navigation to the parent directory.

c. Directory Hierarchy: The directory system organizes files and subdirectories in a hierarchical structure. Subdirectories can contain their own files and subdirectories, creating a tree-like structure. This hierarchy allows for logical organization and easy navigation of files and directories

d. Directory Path: A directory path is a string that specifies the location of a file or directory within the hierarchy. It is typically represented as a series of directory names separated by slashes, such

File System

e. Directory Navigation: Users and applications navigate the directory system to access files and directories. Common directory-related operations include listing the contents of a directory, creating new directories or files, moving or renaming entries, and deleting files or directories.

f. Directory Operations: The file system provides a set of directory-related operations, including creating directories (mkdir), removing directories (rmdir), renaming files or directories (mv), and listing directory contents (ls or dir). These operations are essential for managing the directory system.

For Milestone One, the root directory should be created and initialized with at least the "." and ".." entries. The root directory acts as the starting point for all file and directory management within the volume.

5. A table of who worked on which components

Team member	Assigned Components
Bisum Tiwana	Root Directory
Babak Milani	Free Space Management
Gurpreet Natt	Volume Control Block (VCB)
Mozhgan Ahsant	Documentation and PDF

In this table:

● Gurpreet Natt was responsible for working on the Volume Control Block (VCB).

● Babak Milani was responsible for implementing Free Space Management, including initialization and allocation procedures.

● Bisum Tiwana was in charge of creating and initializing the Root Directory, ensuring it includes the `"."` and `".."` entries.

● Mozhgan Ahsant took care of documenting the project and creating the PDF submission, including the cover page with team information.

6. How did your team work together, how often you met, how did you meet, how did you divide up the tasks.

File System

collectively working on each task during our group meetings, rather than dividing the tasks among us. This approach was feasible because we maintained regular online meetings ever since this milestone was assigned. Our workflow involved one team member sharing their screen with the code visible, while the rest of the team contributed ideas, suggestions, and even shared code through Discord chat. This approach proved highly effective in fostering collaboration among all team members and ensuring a shared understanding of the current milestone task.

7. A discussion of what issues you faced and how your team resolved them.

Challenge: Initially, we encountered the dilemma of where to store our Directory Entry and VCB struct.

Resolution: Our initial plan was to place these structs in mfs.h, but after a thorough discussion, we decided to take a different approach. We opted to create separate header files for these components. The VCB object was maintained as an external global object to facilitate its initialization in fsinit.c. Likewise, we did the same for the Bitmap. We created both the header file and the corresponding C file to facilitate the implementation of bit manipulation functions.

Challenge: Despite including the header files, we encountered an "undefined reference" error for our bitmap functions.

Resolution: To address this issue, we had to make modifications to the makefile to ensure the inclusion of the BitMap object. Specifically, we added BitMap.o as an object to resolve the "undefined reference" error.

Throughout our project on GitHub, we've encountered some challenges, particularly when it comes to managing branches and pull requests. We've found that to overcome these challenges, it's essential for us, as a team, to be well-acquainted with Git workflows. Our collective efforts are key to reviewing each other's pull requests thoroughly. We take this opportunity to provide constructive feedback and openly address any questions or uncertainties regarding code changes. This personalized approach fosters a sense of unity and cooperation within our team, ultimately leading to the successful advancement of our project while maintaining high code quality.